Remote Method Invocation

RMI – Remote Method Invocation

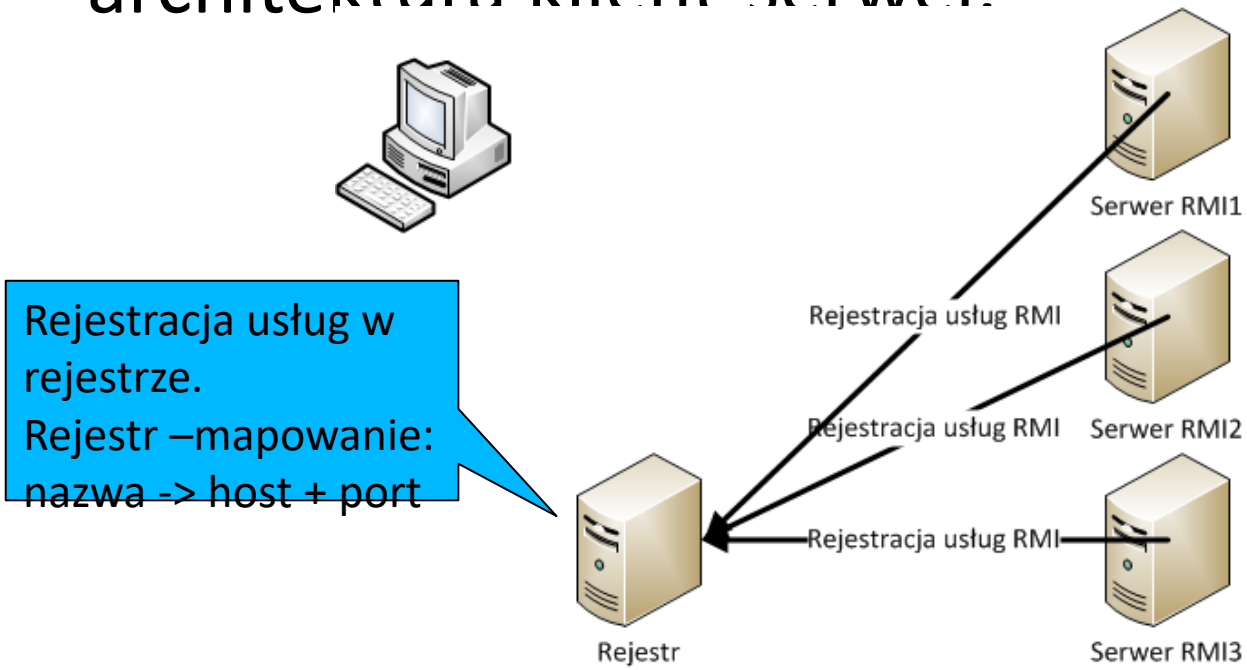
RMI – mechanizm zdalnego wywoływania metod.

RMI implementuje mechanizm Klient-Serwer tak iż na serwerze tworzona jest instancja obiektu, a klient dysponuje jedynie namiastką obiektu, która pozwala mu na wywoływanie metod na zdalnym obiekcie.

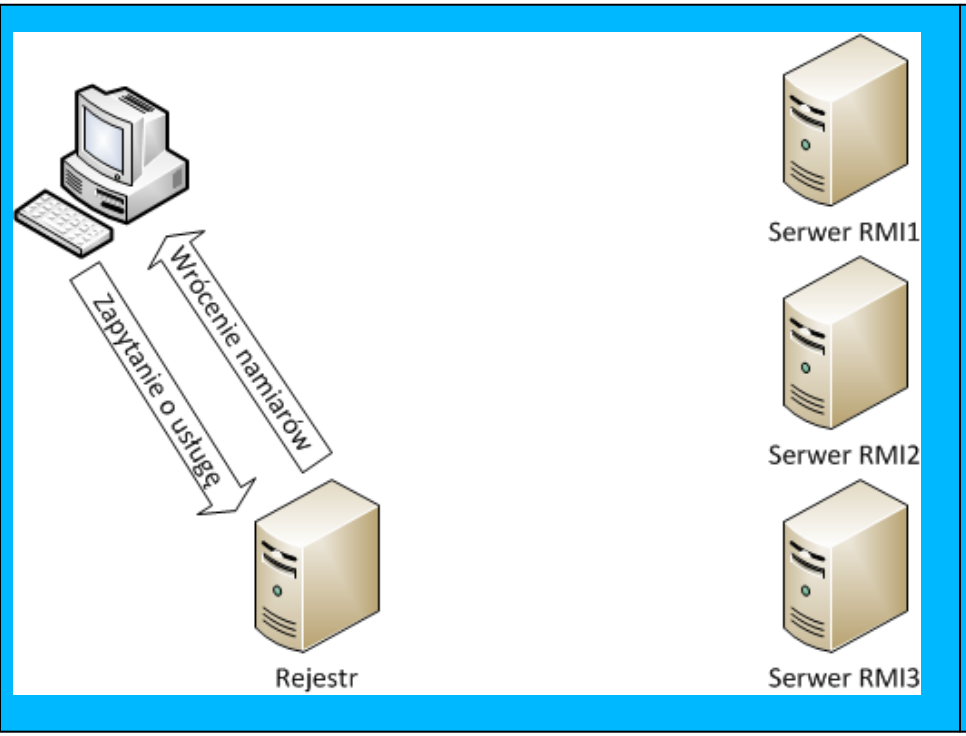
RMI – remote method invocation

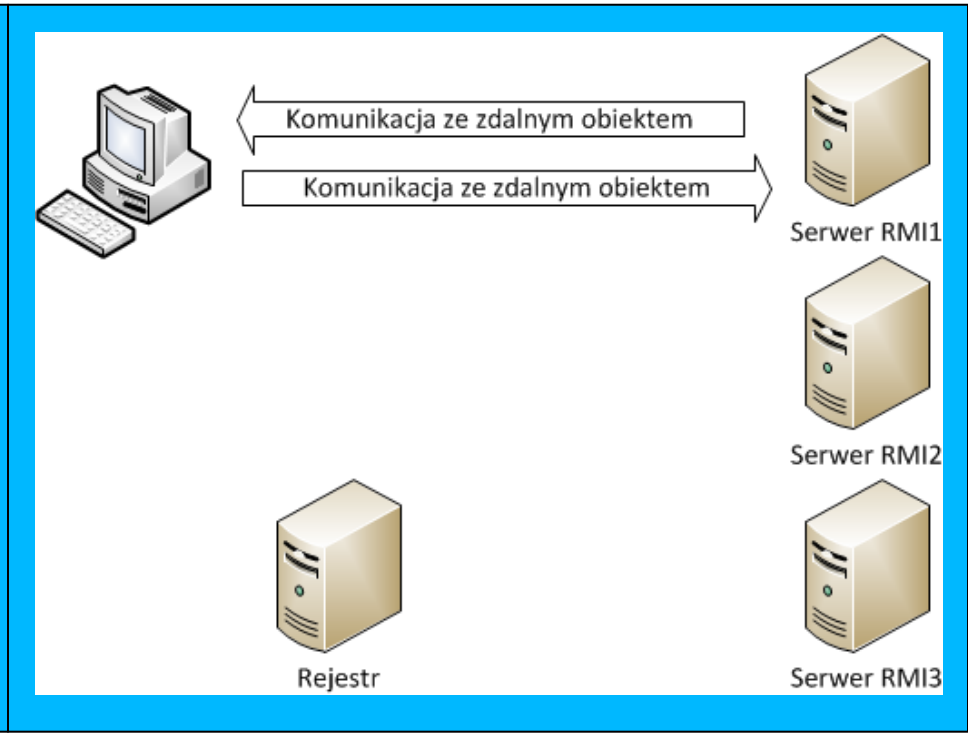
- RMI - Zdalne wywoływanie metod

Pozwala na wywoływanie metod obiektu znajdującego się na zdalnym komputerze- > architektura klient-serwer



RMI


Serwer RMI1
Serwer RMI2
Serwer RMI3
Wrócenie namiarów
Zapytanie o usługę
Rejestr


Komunikacja ze zdalnym obiektem
Komunikacja ze zdalnym obiektem
Serwer RMI1
Serwer RMI2
Serwer RMI3
Rejestr

RMI – tworzenie zdalnego obiektu

1) Stworzenie interfejsu, obsługiwanego przez zdalny obiekt

```
public interface ZdalnyObiektA extends Remote{
    void wyslij(String s) throws RemoteException;
}
```

Metoda obsługiwana przez zdalny obiekt

2) Implementacja interfejsu -> stworzenie klasy

```
public class ZdalnyObiektAImpl implements ZdalnyObiektA{

    @Override
    public void wyslij(String s) throws RemoteException {
        System.out.println(s);
    }

}
```

RMI – implementacja serwera

RMI – implementacja klienta


```
ic class Clinet {
    ublic static void main(String[] args) throws Exception {
        if (System.getSecurityManager()==null){
            System.setProperty("java.security.policy", "policy
            System.setSecurityManager(new SecurityManager());
        }
        String nazwa = "usluga";
        Registry registry = LocateRegistry.getRegistry(4444);
        ZdalnyObiektA service = (ZdalnyObiektA)registry.lookup(nazwa);
        System.out.println("Podłączony");
        service.wyslij("Hallo server");
    }
}
```

RMI - Uwagi

- Uwaga – konieczne jest załadowanie plików polis bezpieczeństwa zarówno dla serwera jak i dla klienta
- Plik polis określa uprawnienia wywołania kodu zarówno na kliencie jak i na serwerze.
- Przykładowy plik polisy:

Pełne uprawnienia na wszystko

```
grant {
    permission java.security.AllPermission;
};
```

java -Djava.security.policy=plik_polis_bezpieczeństwa Serwer
java -Djava.security.policy=plik_polis_bezpieczeństwa Klient

RMI - Uwagi

- Możliwość ustawienia polis bezpieczeństwa bezpośrednio w kodzie

```
if (System.getSecurityManager()==null){
    System.setProperty("java.security.policy", "policy.txt");
    System.setSecurityManager(new SecurityManager());
}
```

  - Polisy bezpieczeństwa musza być ustawione przed włączeniem *SecurityManagera*
  - Polecenie *System.setProperty(key,val)* – pozwala programowo ustawić parametry konfiguracyjne środowiska. Inna opcja to przy uruchamianiu

java –Djava.security.policy=policy.txt

Implementacja interfejsu Remote

- Jeśli RMI ma obsługiwać wielu klientów, to każde połączenie obsługiwane jest przez osobny wątek
- Posiadamy jedną instancję klasy, więc musimy pamiętać o synchroniczności dostępu do pól obiektu

```
public interface ZdalnyObiektB extends Remote{
    void wyslij(String s) throws RemoteException;
    String get() throws RemoteException;
}
public class ZdalnyObiekBtImpl implements ZdalnyObiektB{
    String s;
    @Override
    public synchronized void wyslij(String s) throws RemoteException {
        this.s = s;
    }

    @Override
    public synchronized String get() throws RemoteException {
        return s;
    }
}
```